Vision-Guided State Estimation and Control of Robotic Manipulators Which Lack Proprioceptive Sensors

Vision-guided state estimation and control of robotic manipulators which lack proprioceptive sensors

Valerio Ortenzi[1] Naresh Marturi[1,2] Rustam Stolkin[1] Jeffrey A. Kuo[3] Michael Mistry[4]

***Abstract*— This paper presents a vision-based approach for estimating the configuration of, and providing control signals for, an under-sensored robot manipulator using a single monocular camera. Some remote manipulators, used for decommissioning tasks in the nuclear industry, lack proprioceptive sensors because electronics are vulnerable to radiation. Additionally, even if proprioceptive joint sensors could be retrofitted, such heavy-duty manipulators are often deployed on mobile vehicle platforms, which are significantly and erratically perturbed when powerful hydraulic drilling or cutting tools are deployed at the end-effector. In these scenarios, it would be beneficial to use external sensory information, *e.g.* vision, for estimating the robot configuration with respect to the scene or task. Conventional visual servoing methods typically rely on joint encoder values for controlling the robot. In contrast, our framework assumes that no joint encoders are available, and estimates the robot configuration by visually tracking several parts of the robot, and then enforcing equality between a set of transformation matrices which relate the frames of the camera, world and tracked robot parts. To accomplish this, we propose two alternative methods based on optimisation. We evaluate the performance of our developed framework by visually tracking the pose of a conventional robot arm, where the joint encoders are used to provide ground-truth for evaluating the precision of the vision system. Additionally, we evaluate the precision with which visual feedback can be used to control the robot's end-effector to follow a desired trajectory.**

Fig. 1. A BROKK robot, equipped with a gripper, being used for a pick and place task at the Sellafield nuclear site in UK. This robot has no sensors for measuring joint angles, and so must be directly controlled by a human operator, who pushes a separate switch for each joint and judges the robot's pose by eye. The human operator can be seen controlling the robot from behind a 1.6m thick lead glass window which shields against radiation. For more examples, refer to www.sellafieldsites.com/solution/decommissioning/.

I. Introduction

In several nuclear sites in the UK, as well as important nuclear sites world-wide, such as ongoing work at the Fukushima Daiichi nuclear disaster site, very rugged remote manipulators are used, which lack proprioceptive joint angle sensors. It is not considered feasible to retrofit proprioceptive sensors to such robots: firstly, electronics are vulnerable to gamma and beta radiation; secondly, for nuclear applications, the installation of new sensors on trusted machinery would compromise long-standing certification; thirdly, such robots are predominantly deployed on a mobile base platform and typically use powerful hydraulic drilling and cutting tools at the end-effector. Even if the robot had proprioceptive sensors, such tools cause large and frequent perturbations to the base frame, so that proprioceptive sensors would still be unable to obtain the robot pose with respect to a task frame set in the robot's surroundings. For these reasons, the adoption of external sensors, such as cameras, offers a means of closing the control loop with quantitative feedback, enabling advanced trajectory control and increased autonomy which are currently not possible.

At present, these kinds of remote manipulator machines used for decommissioning tasks on nuclear sites are simply tele-operated, where a human operator controls each joint of the robot individually using a teach pendant or a set of switches, as in Fig. 1. In some cases the human operator controls the robot via CCTV cameras or from behind thick lead glass windows, with very limited depth perception and situational awareness. In other cases, the operator must be positioned near to the robot, wearing protective clothing and breathing apparatus. This not only means that task performances are sub-optimal, but also that humans are being exposed to risk in hazardous environments. This paper contributes a step towards increased autonomy of such tasks, including precise automatic control which is currently almost completely lacking in such industries. Removal of the human factor from such environments could improve safety, and also improve the speed and precision of task performance (*e.g.* scabbling, where an arm must move a grinding tool across a wall or floor to a precise depth, in order to remove contaminated surface material).

This paper shows how reliable feedback of robot configurations can be achieved by tracking several parts of the

School of Engineering[1] and School of Computer Science[4], University of Birmingham, UK.
[2]Kuka Robotics UK Ltd.
[3]National Nuclear Laboratory (NNL) Ltd., UK.
Contact email: vxo344@bham.ac.uk

robot in monocular camera images. We present a framework whose main component estimates the robot configuration by enforcing the equality between a set of transformation matrices relating frames set in the camera, world and the tracked robot parts. For this purpose, we use a model-based vision approach, derived from virtual visual servoing (VVS), in order to track various parts of the robot [1]. The state of the robot, *i.e.* the joint configuration, is estimated by combining this tracked information with the robot's kinematic model. In the following we use the terms state estimation and configuration estimation interchangeably. To solve this estimation problem, we present and compare two different types of non-linear optimisation scheme. In addition to estimating the robot state, we also show how these estimations can be successfully used as quantitative feedback to a classical kinematic controller, in order to make the robot achieve a desired end-effector position.

The vast majority of robots in research labs worldwide possess joint encoders, which is why the vast majority of previous robotics literature (including the visual servoing literature) assumes knowledge of joint angles. In contrast, we believe our approach of estimating the robot configuration by using only monocular camera images, represents both novelty and substantial usefulness for robotics applications in harsh environments. Additionally, the methods proposed in this paper may have wider applications to other problems, e.g.: human-robot or robot-robot interaction; articulated robot calibration; use of a remote camera for servoing of mobile manipulator platforms with respect to surrounding objects.

The remainder of the paper is organised as follows. Sec. II explains our contribution in the context of other related work. Sec. III describes the proposed vision-based configuration estimation scheme along with the details of each component. Sec. IV reports the results of experiments to i) measure the accuracy of our vision-based state estimates, and ii) to measure the precision with which our vision-based approach can be used to control a robot to move its end-effector to a desired position. Sec. V provides concluding remarks and suggests directions for future work.

II. Related Work

The main goal of this work is to provide reliable quantitative configuration feedback for under-sensored robots, so neither the particular choice of visual tracking algorithm nor the particular choice of visual servoing controller form the primary focus of our contribution. This motivates our choice of architecture, which is composed of three separate components: visual tracking of parts of the robot; state estimation; and a controller. We concentrate on state estimation, and choose available methods for the other components. The modularity of this architecture enables flexibility by allowing modification of each of these components independently.

Although deeply influenced by the visual servoing (VS) and articulated body tracking literatures, in this work we are neither interested in controlling the robot directly using visual features, as in a VS paradigm, nor solely in visual tracking of the body of the robot. The VS literature predominantly relies on accurately knowing robot states derived from joint encoders, which are not available in our case. Furthermore, this work endeavours to find a balance between computational speed, performance accuracy, robustness to real-world conditions, and monetary cost. The choice of a single monocular camera represents an efficient and robust solution in terms of cost and reliability in nuclear or other extreme environments, where variable range, strong variations in lighting, reflective surfaces, outdoor sunlight conditions, or dust can cause the performance of many kinds of depth sensors to deteriorate.

Previously, Marchand et al. [2] demonstrated an eye-to-hand visual servoing scheme to control a robot with no proprioceptive sensors. In order to compute the Jacobian of the manipulator, they need to estimate the robot configuration. Thus, they feed the end effector position to an inverse kinematics algorithm for the non-redundant manipulator. Our work is related to this approach, but overcomes the redundancy problem by simultaneously tracking multiple parts of the robot, consequently having more relationships constraining the configuration, making our method potentially applicable to high DOF robots. In [3], a model-based tracker was presented to track and estimate the configuration as well as the pose of an articulated object. In this work, an extended Kalman filter was used to update the object configuration using tracked feature locations. Our approach is marginally related to this work. However, the major differences are that we simultaneously track entire 3D models of various parts of an articulated object and separately define an optimisation problem to estimate the joint values using the tracked poses.

Our main objective is to estimate the robot joint configuration, a problem which is also related to pose estimation. Pose estimation is classically defined for single-body rigid objects, with 6 degrees of freedom (DOF). On the other hand, articulated objects are composed of multiple rigid bodies and possess higher DOF (often redundant). There are also a number of kinematic (and potentially dynamic) constraints that bind together the bodies belonging to kinematic chains. Further, these constraints can also be used to locate and track the chain of robot parts. In this work, for the sake of modularity, the kinematic constraints are used only when estimating the joint values, and not for visual tracking of robot parts, *i.e.* each part of the robot is tracked individually, and then a separate stage of our architecture performs best-fitting of the robot kinematics to the tracked part positions.

A variety of ways to track articulated bodies can be found in [3]–[7]. In contrast to our work, these authors mainly focused on localising parts of the articulated bodies in each image frame, and not on the estimation of joint angles between the connected parts. Additionally, much of this work focussed on tracking parts of robots, but made use of information from the robot's joint encoders to do so, in contrast to the problem posed in our paper.

A real-time system to track multiple articulated objects using RGB-D and joint encoder information is presented in [8]. A similar approach was used in [9] to track and

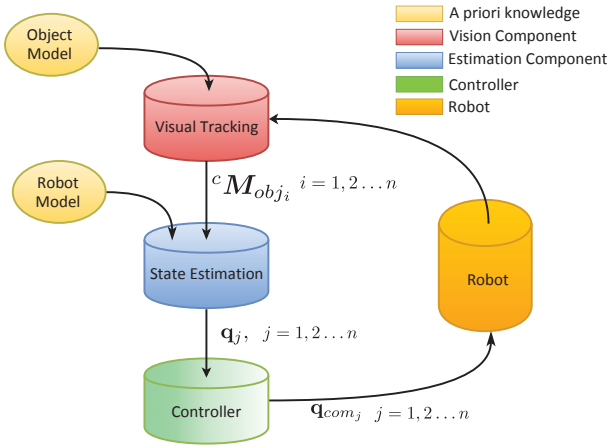


Fig. 2. Overview of our modular pose estimation architecture. The visual tracking module uses RGB images provided by a camera, together with the models of a few selected robot parts, and returns the pose of those parts with respect to the camera frame. The part poses are then used by the state estimation module along with the robot kinematic model, which returns the robot joint configuration. Finally, the controller utilises the overall estimated state to servo the robot.

estimate the pose of a robot manipulator. SimTrack [10] is also a framework for real-time robot tracking using cameras, a Kinect and the joint angles. In [11], a marker-tracking method was used to identify the joint origins of robots. Other notable examples can be found in [12] and [13], where the authors propose to use depth information for better tracking of objects. Recently, an approach based on regression forests has been proposed to directly estimate joint angles using single depth images in [14]. Additionally, in the context of our work, it is worth mentioning some of the human hand pose tracking methods presented in *e.g.* [15]–[17]. However, most of these methods require either posterior information (*e.g.* post-processing of entire image sequences offline to best-fit a set of object poses), or require depth images, or must be implemented on a GPU to achieve online tracking. In contrast, our approach does not make use of depth information, does not require visual tracking of the entire robot, and does not require special markers to be attached to the robot. Instead, a small set of the robot's parts are tracked to estimate the joint configuration. We detail the choice of these robot parts in Sec. III-B.

In summary, the use of depth information alongside standard RGB images can improve the tracking performances. However, it also increases the computational burden and decreases robustness in many real-world applications. Our choice of using only a simple, monocular 2D camera is motivated by cost, robustness to real-world conditions, and also in an attempt to be as computationally fast as possible.

III. State Estimation and Control Framework

As stated earlier, our framework is composed of three main parts. The first component is visual tracking of individual robot links. The model-based visual tracker adopted in this paper is given the 3D models of a small number of selected robot parts, tracks the corresponding poses of these parts and returns the homogeneous transformation matrices between the camera and the tracked objects, ${}^{C}\mathbf{M}_{obj_i}$. Previously, such methods were used for virtual reality [18] and part assembling [19]. The second component makes use of these matrices in estimating the robot's state *i.e.,* the joint configuration $\boldsymbol{q}$. We propose two alternative methods to accomplish this task, both based on optimisation. Finally, we implement a classical kinematic controller to show how these estimations can be used as feedback in a closed-loop control scheme. As previously discussed, this choice of architecture is motivated by the intention of being as modular as possible, *i.e.* the proposed state estimation method can be easily replaced by another one, with no major modifications. The same applies to the visual part tracking module and the controller module. Fig. 2 illustrates the architecture of our proposed framework.

A. Visual Tracking

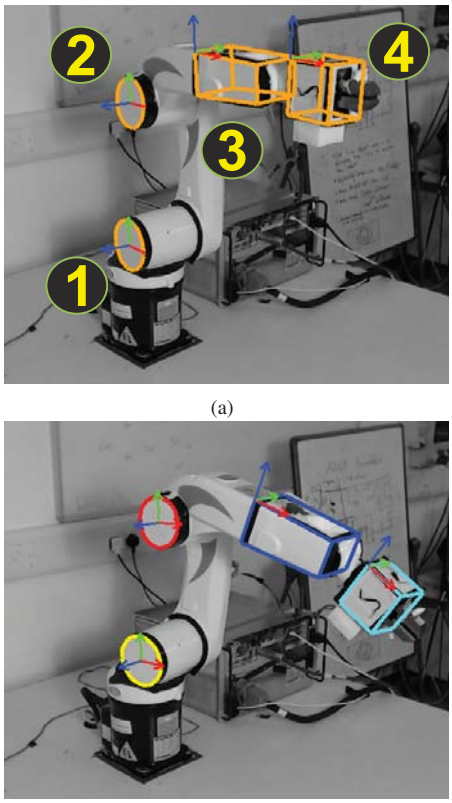
In this work, visual tracking of various parts of the robot has been accomplished using a model-based tracker available in ViSP [20], which projects CAD models of the parts onto camera images. Real-time tracking and pose estimation is achieved by using a Virtual Visual Servoing (VVS) framework [1]. Previous results [18], [19] suggest that such trackers are robust to lighting intensity variations and partial occlusions. Furthermore, this approach runs in real-time on an ordinary CPU without needing GPU acceleration.

Tracking 3D models of robot parts in images is related to the classical pose estimation problem. The underlying idea is to obtain a camera pose for which a projection of the 3D model best fits with the 2D image contours of the robot part. This process involves estimating a rigid transformation between the camera frame and the tracked object frame, ${}^{C}\mathbf{M}_{\mathbf{obj_i}}$. The key steps include: projecting the model using an initial pose estimate (typically the pose estimated at the previous frame), perform a 1D search along the model edges to update the pose, and propagate the updated pose to the next frame. In general, the pose matrix ${}^{C}\mathbf{M}_{obj_i}$ links the 3D object features $\mathbf{P}$ in the world frame to their corresponding projections $\mathbf{p}$ in the image. Assuming the camera intrinsic parameters $\mathbf{K}$ are known, this relationship is given by:

$$\mathbf{p} = \mathbf{K}{}^{\mathbf{C}}\mathbf{M}_{\mathbf{obj_i}}\mathbf{P} \tag{1}$$

Next, it is possible to estimate the transformation parameters by minimising the error $\boldsymbol{\Delta}$ between the current values $\mathbf{s}(\mathbf{r})$, obtained by forward projection of the robot part model using the pose $\mathbf{r}$, and the edges $\mathbf{s}^*$ detected in the image. Its minimisation then corresponds to the movement of a virtual camera (associated with the model) by updating $\mathbf{r}$. The regulation of $\boldsymbol{\Delta}$ requires linking temporal variations of $\mathbf{s}(\mathbf{r})$ with the velocity screw of the virtual camera defined by pose $\mathbf{r}$. This is achieved by using an image Jacobian matrix $\mathbf{J_s}$. This algorithm is based on classical visual servoing, thus we refer the reader to [21] for further details.

In this work, for proof of principle, we tracked four different parts of a KUKA KR5sixx robot, Fig. 3. Trackers are initialised for each part by the user mouse-clicking on corresponding parts in the first image, while the robot is in



(a)

(b)

Fig. 3. Illustration of model-based tracking, using a KUKA KR5sixx robot for proof of principle. (a) Automatic initialised poses in the first frame. Numbers in circles represent the order of the parts selected for this work. (b) Tracked parts in a later frame.

its home position, Fig. 3(a). In the case of highly cluttered scenes the trackers' performance can become erratic due to redundant edges detected in the images. In order to minimise these phenomena, we use a Kalman Filter (KF) to predict and update the final pose after VVS, thus smoothing the changes and also reducing the reactiveness of the tracker. Specifically, we converted the ${}^{C}\mathbf{M}_{\mathbf{obj_i}}$ into pose vectors, *i.e.* tuples of six values, three for the orientation and three for the translation. These values are regarded as the states in the KF. We treat the pose from the estimated ${}^{C}\mathbf{M}_{\mathbf{obj_i}}$ as noisy measurements and update the states, consequently filtering brisk changes. Finally, these updated ${}^{C}\mathbf{M}_{\mathbf{obj_i}}$ from KF are supplied to the next module of our architecture *i.e.* state estimation.

B. State Estimation

In the following, the standard convention for symbols associated with the kinematics of the robot is observed, *e.g.* we define $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ as the configuration and the velocity respectively of the robot in joint space.

For state estimation, we use the following key idea. As shown in Fig. 4, there are two paths from the camera reference frame ${}^{C}RF$ (in yellow) to each tracked part frame ${}^{obj_i}RF$ (in red). As stated, we track four different parts of the robot *i.e,* $i = 1 \ldots 4$. These two paths kinematically coincide,

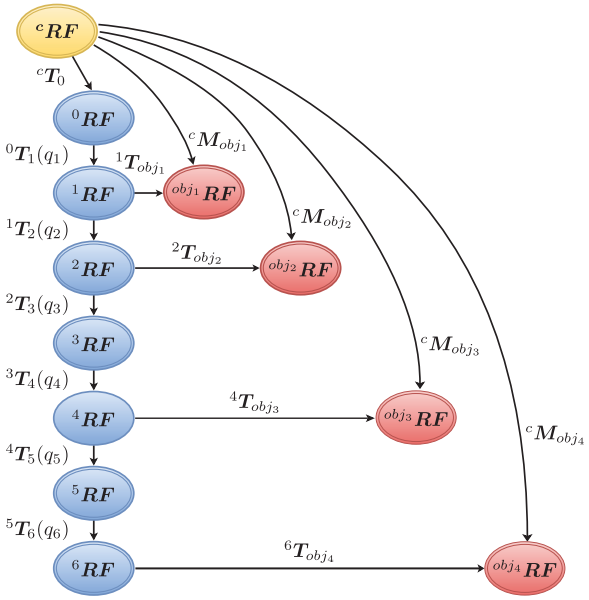


Fig. 4. Illustration of the proposed state estimation model. Nodes represent reference frames and are classified by various colours: camera frame in yellow, robot frames in blue and tracked object frames in red. The two paths leading to each tracked object frame ${}^{obj_i}RF$ from the camera reference frame ${}^{C}RF$ can be seen.

thus we enforce the following equalities to estimate the state:

$$ {}^{C}\mathbf{M}_{obj_i} = {}^{C}\,\boldsymbol{T}_0\ {}^{0}\boldsymbol{T}_{obj_i}(\boldsymbol{q}) \tag{2} $$

where, ${}^{C}\boldsymbol{T}_0$ is the transformation from camera to world frame and ${}^{0}\boldsymbol{T}_{obj_i}(\boldsymbol{q})$ represents the transformation from world to object i frame parametrised over the joint values $\boldsymbol{q}$, *i.e.*, ${}^{0}\boldsymbol{T}_{obj_i}(\boldsymbol{q})$ embeds the kinematic model of the robot. Specifically, for each tracked robot part, we get:

$$ {}^{C}\mathbf{M}_{obj_1} = {}^{C}\boldsymbol{T}_0\ {}^{0}\boldsymbol{T}_1(q_1){}^{1}\boldsymbol{T}_{obj_1} \tag{3} $$

$$ {}^{C}\mathbf{M}_{obj_2} = {}^{C}\boldsymbol{T}_0\ {}^{0}\boldsymbol{T}_1(q_1){}^{1}\boldsymbol{T}_2(q_2){}^{1}\boldsymbol{T}_{obj_2} \tag{4} $$

$$ {}^{C}\mathbf{M}_{obj_3} = {}^{C}\boldsymbol{T}_0\ {}^{0}\boldsymbol{T}_1(q_1){}^{1}\boldsymbol{T}_2(q_2){}^{2}\boldsymbol{T}_3(q_3){}^{3}\boldsymbol{T}_4(q_4){}^{4}\boldsymbol{T}_{obj_3} \tag{5} $$

$$ \begin{aligned} {}^{C}\mathbf{M}_{obj_4} = {}^{C}\boldsymbol{T}_0\ {}^{0}\boldsymbol{T}_1(q_1)\ {}^{1}\boldsymbol{T}_2(q_2)\ {}^{2}\boldsymbol{T}_3(q_3) \\ {}^{3}\boldsymbol{T}_4(q_4)\ {}^{4}\boldsymbol{T}_5(q_5)\ {}^{5}\boldsymbol{T}_6(q_6)\ {}^{6}\boldsymbol{T}_{obj_4} \end{aligned} \tag{6} $$

The state of the robot is now estimated by imposing the equality given in (2), and casting it as an optimisation problem. As already mentioned, we assume that we know the initial configuration of the robot (occupying its home position in the first image) and its kinematic model. The robot's initial configuration is used as a seed for the first iteration of the optimisation problem and the kinematic model is used to compute ${}^{0}\boldsymbol{T}_{obj_i}(\boldsymbol{q})$. The optimisation problem is then stated as:

$$ \begin{aligned} &\underset{\boldsymbol{q}}{\text{minimise}} \quad \sum_i \boldsymbol{e}_i(\boldsymbol{q}) \\ &\text{subject to} \quad |\boldsymbol{q}_j| \leq \boldsymbol{q}_{max} \end{aligned} \tag{7} $$

where

$$ \boldsymbol{e}_j(\boldsymbol{q}) = vec({}^{C}\boldsymbol{M}_{obj_i} - {}^{C}\,\boldsymbol{T}_0\ {}^{0}\boldsymbol{T}_{obj_i}(q)) \tag{8} $$

represents an error in the difference of the two paths shown in Fig. 4 to define a transformation matrix from the camera frame to the tracked objects frames, and $\boldsymbol{q}_{max}$ is the joint limit for the joint. In order to compute ${}^{0}\boldsymbol{T}_{obj_i}(\boldsymbol{q})$, we use the convention of Denavit-Hartenberg parameters. The overall estimation schema along with the transformation matrices between each reference frame are shown in Fig. 4.

Two different optimisation schemes have been implemented. The trackers return a set of matrices, *i.e.* one for each tracked part. These matrices can be used all together, which we term the "full" method. Alternatively, the sets of equations coming from each of the four ${}^{C}\boldsymbol{M}_{obj_i}$ can be used in series to solve for subsets of joint variables, which we call the "chained" method. From Fig. 4, the following dependencies can be observed for each tracked object: 1) first object's position ${}^{obj_1}RF$ depends only on q_1; 2) second object's position ${}^{obj_2}RF$ on q_1 and q_2; 3) third object's position ${}^{obj_3}RF$ on q_1, q_2, q_3 and q_4; 4) finally fourth object's position ${}^{obj_4}RF$ on all the six joints. As shown in Fig. 3, in this work we track two cylindrical and two cuboid shaped parts of a KUKA KR5sixx robot for proof of principle. However, this choice is not a limitation of our work, and a variety of different parts could be chosen. Nevertheless, the parts must be chosen such that they provide sufficient information about all joints of the robot. Due to the modular architecture followed, using alternative parts for visual tracking will not affect the estimation scheme.

Because the "full" method uses all of the tracked part poses at once, the optimisation problem is as given in (7), with $i = 4$ and $j = 6$. In this case, the solution is the tuple of joint angles that best fits all the equations. Alternatively, the chained method uses each object to estimate only a subset of joint values. These, in turn, are used as known parameters in the successive estimation problems. In the presented example, q_1 can be retrieved using obj_1, as in:

$$\begin{aligned} \underset{q_1}{\text{minimise}} \quad & \boldsymbol{e}_1(q_1) \\ \text{subject to} \quad & |\boldsymbol{q}_1| \leq \boldsymbol{q}_{max} \end{aligned} \tag{9}$$

and from now on it is treated as known. q_2 is estimated using obj_2:

$$\begin{aligned} \underset{q_2}{\text{minimise}} \quad & \boldsymbol{e}_2(q_1, q_2) \\ \text{subject to} \quad & |\boldsymbol{q}_2| \leq \boldsymbol{q}_{max}, \end{aligned} \tag{10}$$

In a similar fashion, q_3 and q_4 are estimated using obj_3:

$$\begin{aligned} \underset{q_3, q_4}{\text{minimise}} \quad & \boldsymbol{e}_3(q_1, q_2, q_3, q_4) \\ \text{subject to} \quad & |\boldsymbol{q}_j| \leq \boldsymbol{q}_{max},\ j = 3, 4. \end{aligned} \tag{11}$$

And finally, obj_4 provides the equations to compute q_5 and q_6:

$$\begin{aligned} \underset{q_5, q_6}{\text{minimise}} \quad & \boldsymbol{e}_4(\boldsymbol{q}) \\ \text{subject to} \quad & |\boldsymbol{q}_j| \leq \boldsymbol{q}_{max},\ j = 5, 6. \end{aligned} \tag{12}$$

There are a number of considerations regarding these two alternative approaches. Using only one object at a time, as in the chained method, the quality of configuration estimation becomes highly dependent on the tracking performance for each individual part. Although it induces the advantage of being robust to single part tracking failure (producing outliers that influence the estimation of only the relative subset of angles), it adds the disadvantage of propagating the possible error of already estimated angles in subsequent estimations. On the other hand, the full method overcomes this problem. However, it has to accommodate a solution for a higher number of equations. Thus, an error in any parameter will potentially lead to estimation errors on all joints, independently of the tracking performance.

C. Controller

For proof of principle, we implemented a classical kinematic controller of the form given in Eq. (13) to validate our methodology and also to demonstrate how the vision-derived state estimations can be used to servo the robot's end-effector to a desired point in the workspace.

$$\dot{\boldsymbol{q}}_{ref} = \boldsymbol{J}^{\dagger}(\boldsymbol{q})(\boldsymbol{K}_P \boldsymbol{e}) - \boldsymbol{K}_D \dot{\boldsymbol{q}} \tag{13}$$

Here, $\dot{\boldsymbol{q}}_{ref}$ is the desired/reference velocity, and $\boldsymbol{J}^{\dagger}(\boldsymbol{q})$ is the pseudo-inverse of the robot Jacobian computed using our estimated joint configuration. The pseudo-inversion is needed since the tasks are positional, thus the Jacobian $\boldsymbol{J}$ is 3x6. The error $\boldsymbol{e}$ has been defined as the difference between desired and estimated Cartesian positions of the end-effector. Note that the Cartesian positions are updated in each iteration using the direct kinematics with the estimated configuration. Finally, $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ are the proportional and derivative gain matrices. The estimation of the robot joint velocity has been computed as the difference between the present and the previous robot configuration. Since the commands sent to the robot are in position, Eq. (13) is integrated numerically to compute such commands:

$$\boldsymbol{q}_{cmd} = \boldsymbol{q}_t + \Delta t\ \dot{\boldsymbol{q}}_{ref} \tag{14}$$

where $\boldsymbol{q}_{cmd}$ are the commands sent to the robot, $\boldsymbol{q}_t$ is the estimated robot configuration, Δt is the integration time and $\dot{\boldsymbol{q}}_{ref}$ is as defined in Eq. (13).

IV. Experimental Results

A. Experimental Setup

The estimation and control framework presented in the previous section has been evaluated in real-time on an experimental set-up comprising a 6 DOF KUKA KR5 robot and a commercial Logitech c920 camera. Even though this robot is equipped with joint encoders, we do not use those values for state estimation or control, but we record them as ground truth which we use for performance evaluation. Our architecture was implemented in C++ and executed on an ordinary PC running Linux (8 GB RAM, 3.1 GHz Intel core i5 CPU). The communication between the PC and the robot controller was realised using TCP/IP. CAD models of robot parts were supplied to the tracker along with the part poses for a pre-defined home position during the initialisation step. All the proposed optimisation problems are solved using the constrained optimisation by linear approximation (COBYLA) algorithm available in NLopt C++ library [22].

Two series of experiments have been conducted. Firstly, we assess the precision of the proposed framework in estimating the robot configuration. For this purpose, the robot was asked to repeatedly perform three different trajectories, and the vision-estimated joint angles were compared to the ground-truth angles derived from the robot's joint encoders. Secondly, we used the vision-derived state estimates as feedback in a kinematic control loop in order to perform regulation tasks. On average, a single control iteration including visual tracking of all the parts takes a computational time of 12 ms, *i.e.* the system is capable of running at equivalent of > 80 frames per second on the CPU of an ordinary computer.

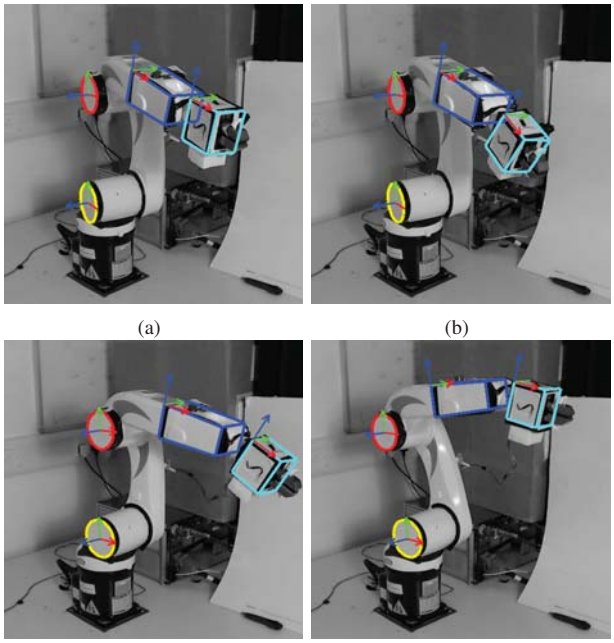
B. State Estimation

To analyse the performance of our state estimation module, 3 arbitrary trajectories were performed as shown in the first column of Table I. The trajectories were selected such that all robot parts were visible in the camera field of view at all times. Trajectory 3 was chosen so as to excite one joint at a time. Fig. 5 shows screenshots of the tracking during various trajectories. More results can be found in the provided supplementary video. Each trajectory was executed 5 times in order to perform quantitative analysis (measuring accuracy and precision). The obtained results are summarised in Table I. It can be seen that the chained method consistently outperformed the full method in all the tests, and its average error typically remains lower than 4° on all joints. Fig. 6(a) and 6(b) show the estimated states of all joints during trajectories 2 and 3, respectively using the chained method. These results clearly demonstrate the efficiency of our state estimation framework in terms of accuracy and repeatability.

C. Controller

In this section, we use the values provided by the state estimation module as feedback for a kinematic controller. Five different goal positions were selected in the robot task space and the objective was to regulate the error using the estimated joint values and position the robot end-effector at the desired location. Fig. 7 reports the evolution of robot end-effector position values during one of the runs. For comparison purposes, we also show the values computed using joint encoders. It can be seen that the process converged at around ± 10 mm on all three axes, which is quite acceptable for the tasks this framework has been designed for. Additionally, the overall task space convergence was analysed using a cost function given by the squared error. Fig. 8 shows the cost variations during all five tasks. Also the trajectories followed by the end-effector during two of the tasks are shown in Fig. 9. The obtained results clearly demonstrate the robustness of our approach in estimating the robot joint state values and using them for regulation tasks.

Additionally, a test was performed in which the robot was required to move its end-effector along a trajectory tracing out the perimeter of a square. Corners of the square were supplied as goal positions to the controller. Results of three different runs are illustrated in Fig. 10.

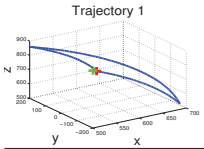
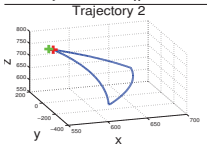
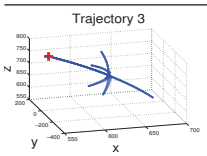


(a) (b) (c) (d)

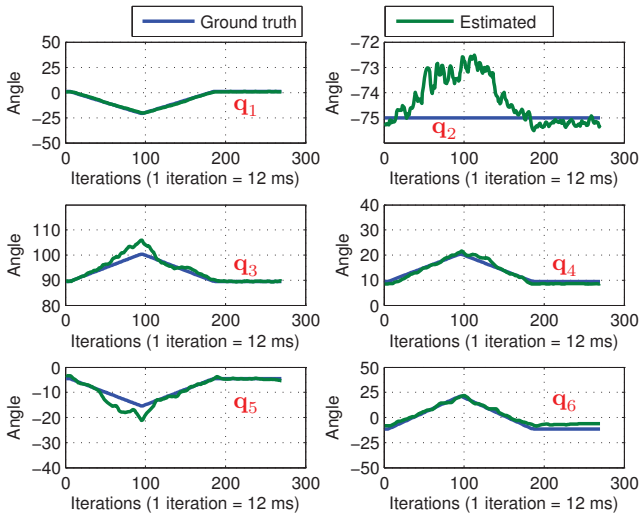
Fig. 5. Tracking of parts for state estimation during various trajectories. More results can be found in the provided supplementary video.

TABLE I

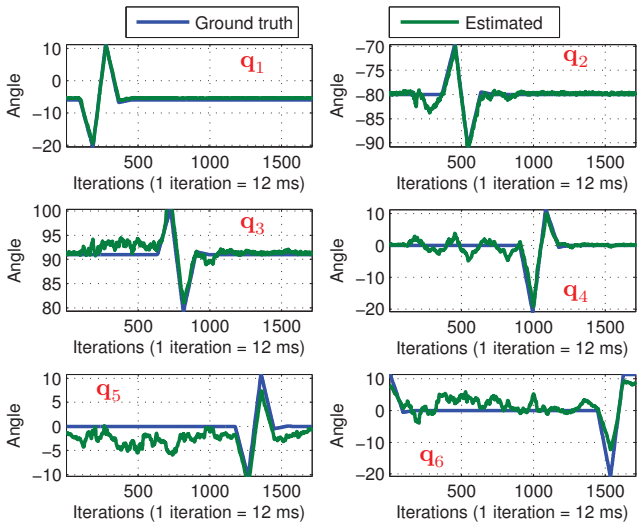
PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK.

Trajectory	Joint	Chained method		Full method	
		RMSE	STD	RMSE	STD
	q_1	**0.9276**	**0.8853**	3.0474	2.8839
	q_2	**2.2015**	1.9099	5.6387	**1.7151**
	q_3	**3.5611**	**1.6257**	5.3594	2.8850
	q_4	**2.3186**	**2.2327**	2.7001	2.3568
	q_5	3.3825	2.4752	**3.0220**	**2.4087**
	q_6	3.8366	3.2841	**3.4607**	**2.6246**
	q_1	**0.4684**	**0.4647**	1.7158	1.6921
	q_2	**1.0553**	**0.8797**	6.5830	1.1917
	q_3	**2.1721**	**1.8104**	6.4424	3.5762
	q_4	**0.9231**	0.8509	1.3283	**0.7681**
	q_5	2.6000	2.0987	**2.2649**	**2.0423**
	q_6	3.4227	**2.0007**	**2.7552**	2.4180
	q_1	**0.5818**	**0.2619**	1.4042	0.9870
	q_2	**0.8010**	**0.7774**	9.7963	1.4538
	q_3	**1.3725**	**1.0542**	10.0045	1.9927
	q_4	**1.5080**	**1.5049**	2.1949	2.0882
	q_5	**2.5129**	1.6345	2.5898	**1.4391**
	q_6	4.1069	2.6367	**3.1573**	**2.5723**
Overall (avg. values)	q_1	**0.6593**	**0.5373**	2.0558	1.8543
	q_2	**1.3526**	**1.1890**	7.3393	1.4535
	q_3	**2.3686**	**1.4968**	7.2688	2.8180
	q_4	**1.5832**	**1.5295**	2.0744	1.7377
	q_5	2.8318	2.0695	**2.6256**	**1.9634**
	q_6	3.7887	2.6405	**3.1244**	**2.5383**

RMSE:- Root mean square error. STD:- Standard deviation.



(a) State estimation for trajectory 2.



(b) State estimation for trajectory 3.

Fig. 6. Real and estimated states with various trajectories using chained method. Angles are expressed in degrees.

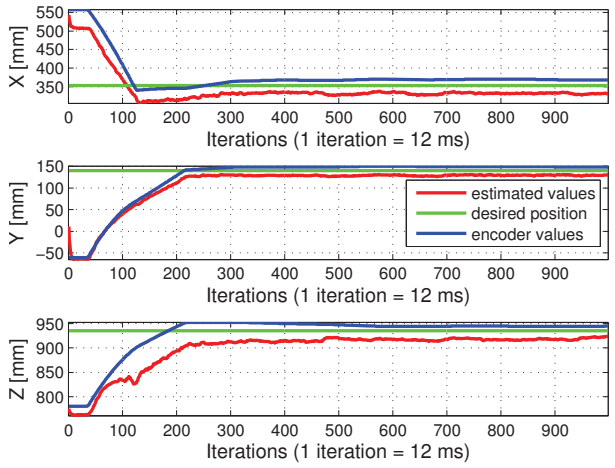


Fig. 7. Evolution of the robot end-effector positional values on all three axes during a regulation task.

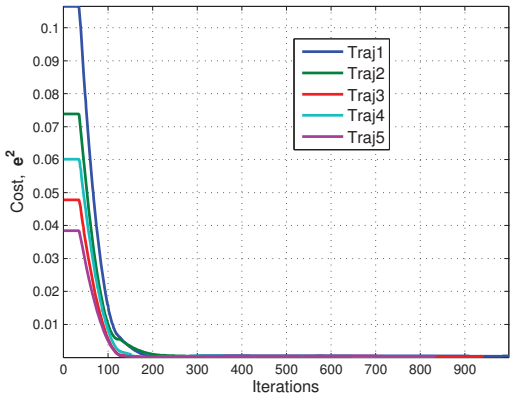
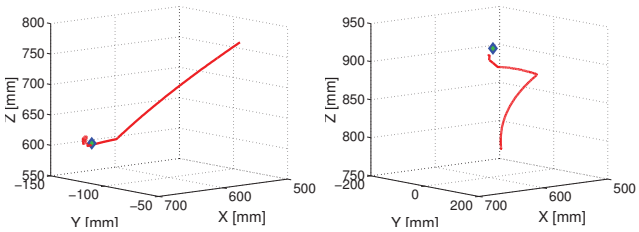


Fig. 8. Cost variations during all five regulation tasks.



(a) Run - 1 (b) Run - 2

Fig. 9. Trajectories followed by the end-effector during two different regulation tasks. Diamond shaped point represents goal position.

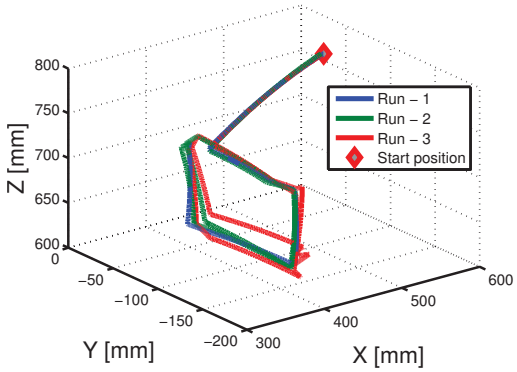


Fig. 10. Square-perimeter trajectories followed by the end-effector. Diamond marker represents robot starting position.

D. Discussion

During these experiments, the root mean square errors of the joint angle estimates with respect to ground truth values were generally less than $4°$. The chained method outperforms the full method most of the time. In our opinion, this can be explained by supposing that the first three robot parts were tracked very accurately and so the chained method was able to estimate the relative joint configurations very precisely. The last robot part (a cuboid section of the gripper mounted on the end-effector) represented more of a challenge to the tracking algorithm, and demonstrated slightly worse results. On the other hand, the full method averages out the performances of the trackers for each robot part, thereby

achieving slightly better performances in the last two joints at the cost of a slightly higher error on the first two joints. In the Cartesian regulation tasks, the errors at pseudo-steady state are approximately 10 mm on each Cartesian axis.

V. Conclusion

This paper has presented a framework to estimate the configuration of an under-sensored robot through the use of a single monocular camera. First, we track several parts of the robot using an algorithm based on virtual visual servoing, then we use the information given by each part's tracker, combined with the kinematic model of the robot, to compute an estimation of the configuration of the robot. We present two alternative methods, and highlight their differences. A study of the precision and robustness of the estimation methods was presented, as well as the results of using those state estimations in a kinematic controller used to perform Cartesian tasks. Joint angle errors not greater than 4° were achieved consistently in the estimation module using the chained method, and a Cartesian error of $10\ \mathrm{mm}$ was achieved while performing Cartesian regulation tasks. These results are acceptable for many practical tasks in the nuclear industry, however in future work, we believe that these estimates can be improved by: incorporating kinematic constraints into the visual tracking module; estimating velocities and accelerations; exploiting such physical information to robustify tracking [23].

This work has been motivated by the needs arising in the nuclear industry, where many robotic devices do not possess proprioceptive sensors. However, we believe that a larger community can benefit from this work. For example, this framework could be used in other fields of application, such as human-robot interaction, where robots ideally are asked to understand the movements of human agents in order to perform safe and effective interaction. In future work, we aim to generalise this framework to other robots and to different tasks. Finally, we are interested in estimating velocities and accelerations alongside joint values.

This framework can also be easily extended to take into consideration possible motions of the base of the robot. Because of forceful interactions of end-effector tools with the environment, the base of mobile manipulators can be significantly perturbed. These unpredictable motions of the base can be regarded as additional degrees of freedom to be modelled in the kinematics of the robot. A possible solution would be an additional tracker for the base.

VI. Acknowledgements

This work was supported by: a UK NDA bursary; Innovate UK KTP partnership 9573; an NNL Research Fellowship; EU H2020 RoMaNS, 645582; EU H2020 CogIMon, 644727; and EPSRC grant EP/M026477/1.

References

[1] E. Marchand and F. Chaumette, "Virtual visual servoing: A framework for real-time augmented reality," in *Proc. Eurographics*, vol. 21(3), Saarebrcken, Germany, September 2002, pp. 289–298.

[2] E. Marchand, F. Chaumette, F. Spindler, and M. Perrier, "Controlling an uninstrumented manipulator by visual servoing." *I. J. Robotic Res.*, vol. 21, no. 7, pp. 635–648, 2002.

[3] K. Nickels and S. Hutchinson, "Model-based tracking of complex articulated objects," *IEEE Trans. Robot. Automat.*, vol. 17, no. 1, pp. 28–36, 2001.

[4] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 7, pp. 932–946, 2002.

[5] A. I. Comport, É. Marchand, and F. Chaumette, "Complex articulated object tracking," in *Articulated Motion and Deformable Objects*. Springer, 2004, pp. 189–201.

[6] A. I. Comport, É. Marchand, and F. Chaumette, "Kinematic sets for real-time robust articulated object tracking," *Image Vision Comput.*, vol. 25, no. 3, pp. 374–391, 2007.

[7] S. Pellegrini, K. Schindler, and D. Nardi, "A generalisation of the icp algorithm for articulated bodies." in *Proc. British machine vision conference*, September 2008.

[8] K. Pauwels, V. Ivan, E. Ros, and S. Vijayakumar, "Real-time object pose recognition and tracking with an imprecisely calibrated moving rgb-d camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 2733–2740.

[9] M. Klingensmith, T. Galluzzo, C. Dellin, M. Kazemi, J. A. Bagnell, and N. Pollard, "Closed-loop servoing using real-time markerless arm tracking," in *IEEE Int. Conf. Robot. Autom. (Humanoids workshop)*, 2013.

[10] K. Pauwels and D. Kragic, "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[11] N. B. Figueroa, F. Schmidt, H. Ali, and N. Mavridis, "Joint origin identification of articulated robots with marker-based multi-camera optical tracking systems," *Robot. Auton. syst.*, vol. 61, no. 6, pp. 580–592, 2013.

[12] T. Schmidt, R. Newcombe, and D. Fox, "DART: Dense Articulated Real-Time Tracking," in *proc. Robotics: Science and Systems*, Berkeley, USA, 2014.

[13] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3143–3150.

[14] F. Widmaier, D. Kappler, S. Schaal, and J. Bohg, "Robot arm pose estimation by pixel-wise regression of joint angles," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2016*. IEEE, May 2016.

[15] D. Park and D. Ramanan, "Articulated pose estimation with tiny synthetic videos," in *Proc. IEEE Int. Conf. Comput. Vision. Pattern Recog. Workshops*, 2015, pp. 58–66.

[16] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, no. 1, pp. 116–124, Jan. 2013.

[17] S. Gammeter, A. Ess, T. Jäggli, K. Schindler, B. Leibe, and L. Van Gool, "Articulated multi-body tracking under egomotion," in *Proc. European Conf. Comp. Vision*. Springer, 2008, pp. 816–830.

[18] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework," *IEEE Trans. Visual. Comput. Graphics*, vol. 12, no. 4, pp. 615–628, 2006.

[19] S. Dembélé, N. Le Fort-Piat, N. Marturi, and B. Tamadazte, "Gluing free assembly of an advanced 3d structure using visual servoing." in *Micromechanics and Microsystems Europe Workshop*, 2012, pp. 6–12.

[20] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robot. Automat. Mag.*, vol. 12, no. 4, pp. 40–52, 2005.

[21] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robot. Automat. Mag.*, vol. 13, no. 4, pp. 82–90, 2006.

[22] S. G. Johnson, "The nlopt nonlinear-optimization package," http://ab-initio.mit.edu/nlopt.

[23] D. J. Duff, T. Mörwald, R. Stolkin, and J. Wyatt, "Physical simulation for monocular 3d model based tracking," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5218–5225.